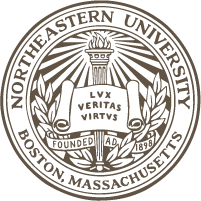
[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiAxdP4_qrZAhUlxoMKHZDACFUQjRwIBw&url=https://www.northeastern.edu/guidelines/resources/index.html&psig=AOvVaw3ZXiw7Od7CjTCeUGCehvMB&ust=1518889630579473)

ADVANCE IN DATA SCIENCE

AND

ARCHITECTURE

Data wrangling Edgar data from text files

SUBMITTED BY

DIVYA PRIYA EMMANUEL

GAURANG DAVDA

NIRANJHANI VASUDEVAN

Table of Contents:

Table of Contents

**Abstract3**

**Section 1- Parse the file3**

Collect the CIK and accession number from the user and create URL to access company filings3

URL creation for the 10q HTML page by traversing URL obtained in Step 1 4

Creating folder name with expected company name 4

Directory and folder creation 4

Function to extract the table6

Find if particular style tag present6

Obtain table data from 10q filling 7

Writing the extracted output in a csv7

Capturing each action in a log file7

Zipping the csv files8

**Section 2 - Dockerize the pipeline4**

**Section 3 – Questions**5

**Abstract**

This report summarizes the implementation of Data Scraping executed on EDGAR website. The following task are performed as part of the assignment :

Section 1 : Explains the web scrapping and extraction of the 10q filing data tables from

EDGAR website with given CIK and accession number of a company and

Section 2 : Illustrates the implementation of docker image that was build to automate

Section 1 task which could be parameterized in a config file and the resultant

Output is hosted on Amazon S3

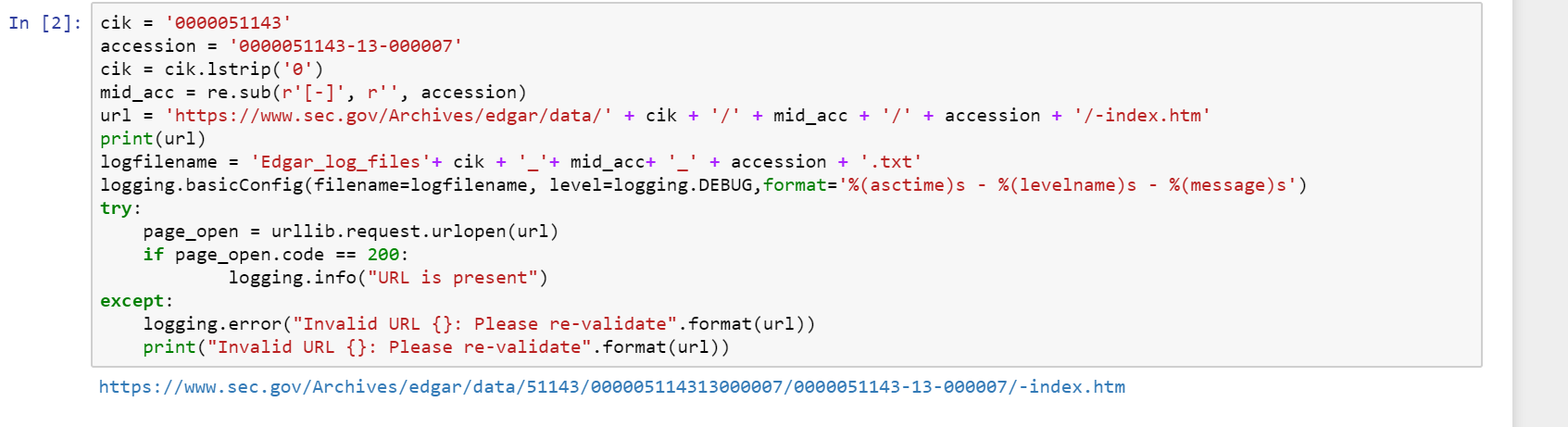
Section 3 : Discussed the output for better understanding

**Section 1 - Parse the file**

**STEP 1: Collect the CIK and accession number from the user and create**

**URL to access company filings**

With the given CIK and accession number by the user, we are programmatically generating the corresponding company CIK. URL that contains the company filling is created successfully. The leading zeros in CIK are eliminated before hitting the required company URL. In order to verify whether the URL is valid, try catch block has been implemented to handle if exception occurred.



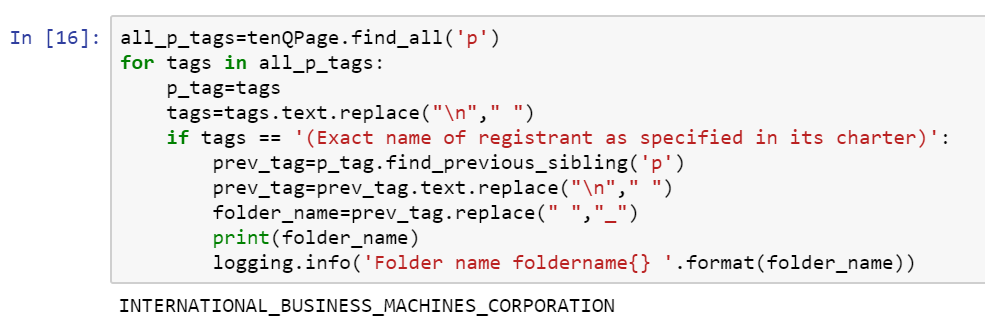
**STEP 2: URL creation for the 10q HTML page by traversing URL obtained**

**in Step 1**

In this step, the HTML page of the 10q filling for the requested company is fetched and parsed using the URL created in Step 1. And the same try catch block from step 1 is performed to validate the generated URL.

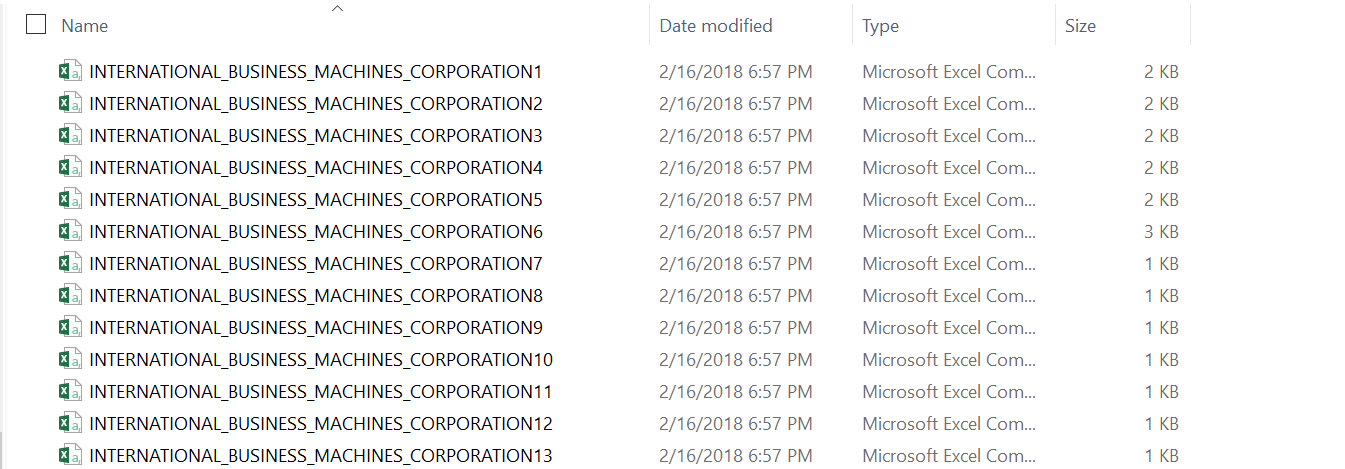
**STEP 3: Creating folder name with expected company name**

This step helps in extracting the company name which will used as the file name for our outputs.

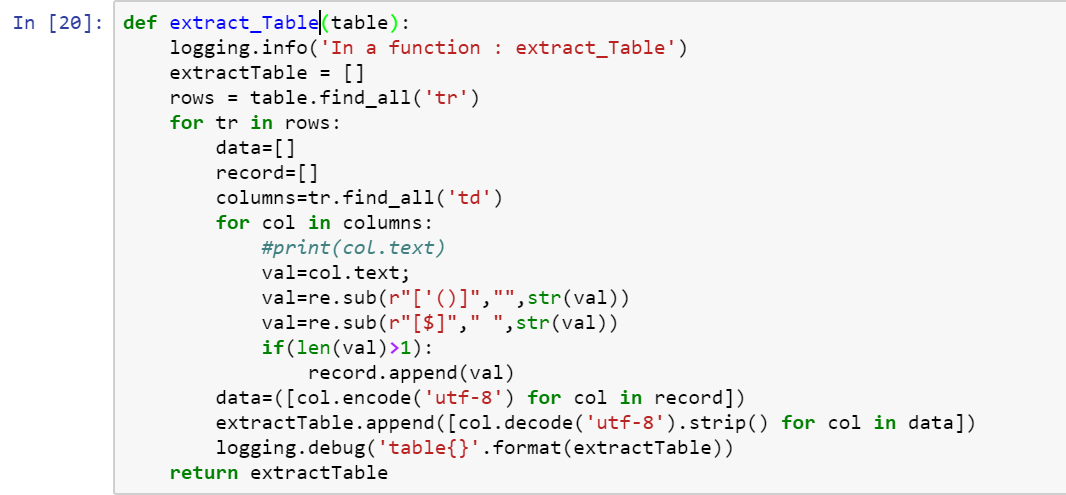


**STEP 4: Directory and folder creation**

Particular directory is created along with a folder wherein all the output csv files will be created.

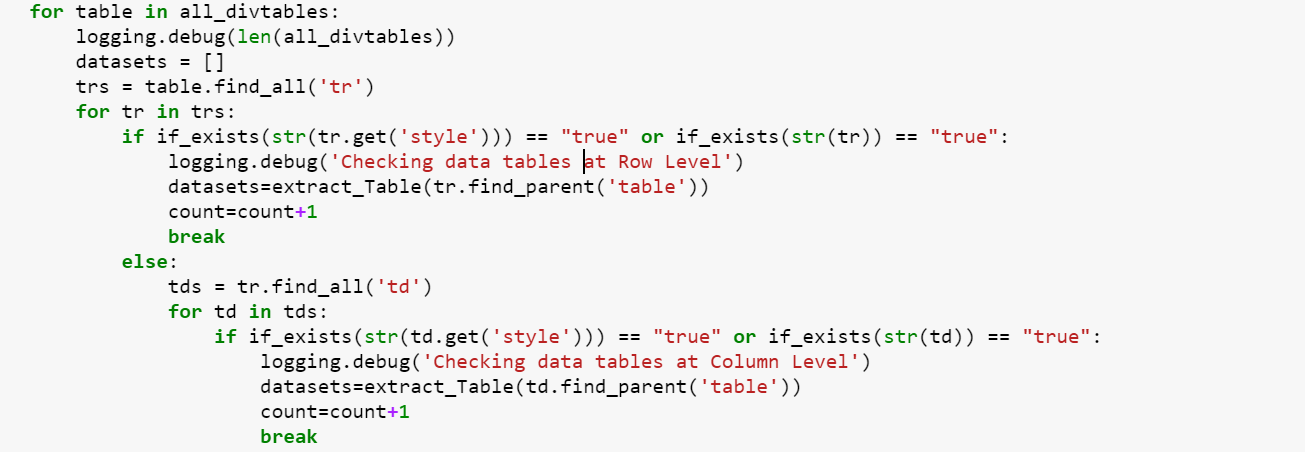


**STEP 4: Function to extract the table**

The function in this steps, helps to read and structure the table from the HTML page. Unwanted symbols during the extraction are removed by replacing with space. Then the obtained data is encoded and decoded to remove all ASCII dumps which are created while parsing the data.

**STEP 5: Find if particular style tag present**

In order to differentiate the table from other data contents in the 10q filling HTML page, we have considered to identify table location by finding the background colour change. Hence, a function is created to recognise the value in each style tag mentioned in the particular HTML page.

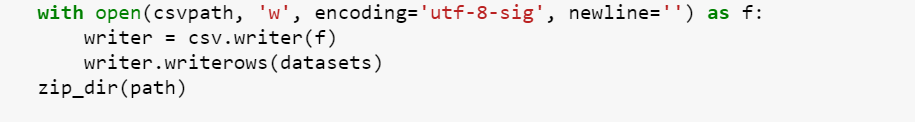


**STEP 6: Obtain table data from 10q filling**

The Actual data extraction from the table is performed in this step with the help of function from Step 5.

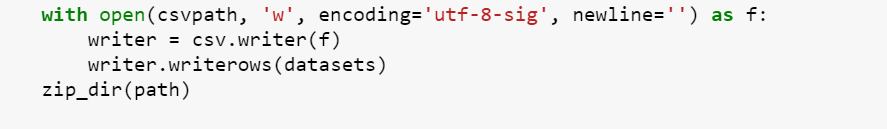
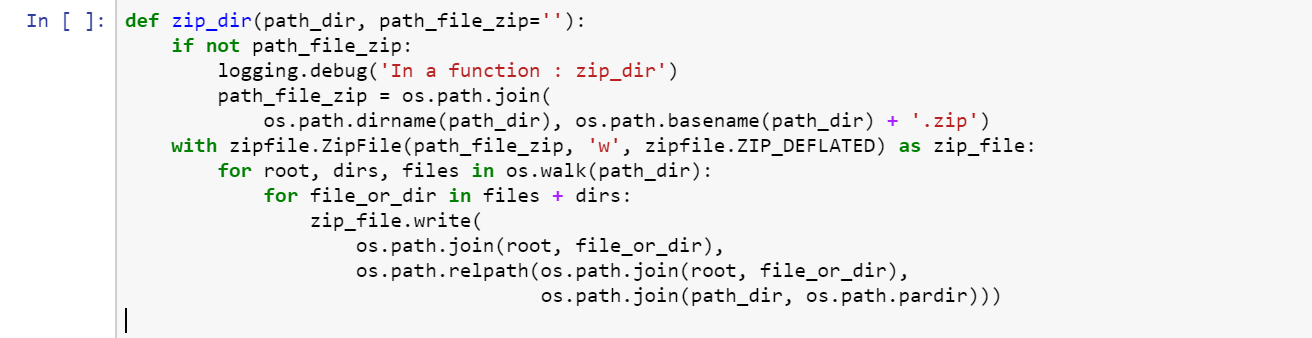
**STEP 7: Writing the extracted output in a csv**

Here each extracted table is written in separate csv files.



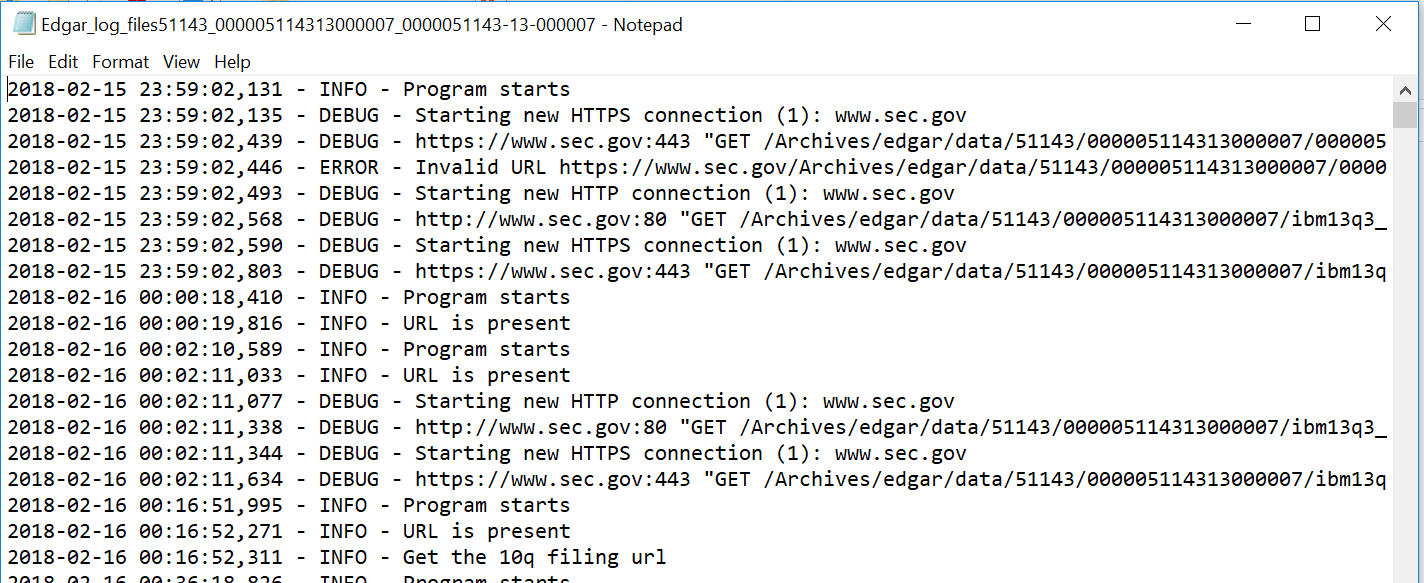
**STEP 8** **: Zipping the csv files**

The extracted csv files are zipped



**STEP 9: Capturing each action in a log file**

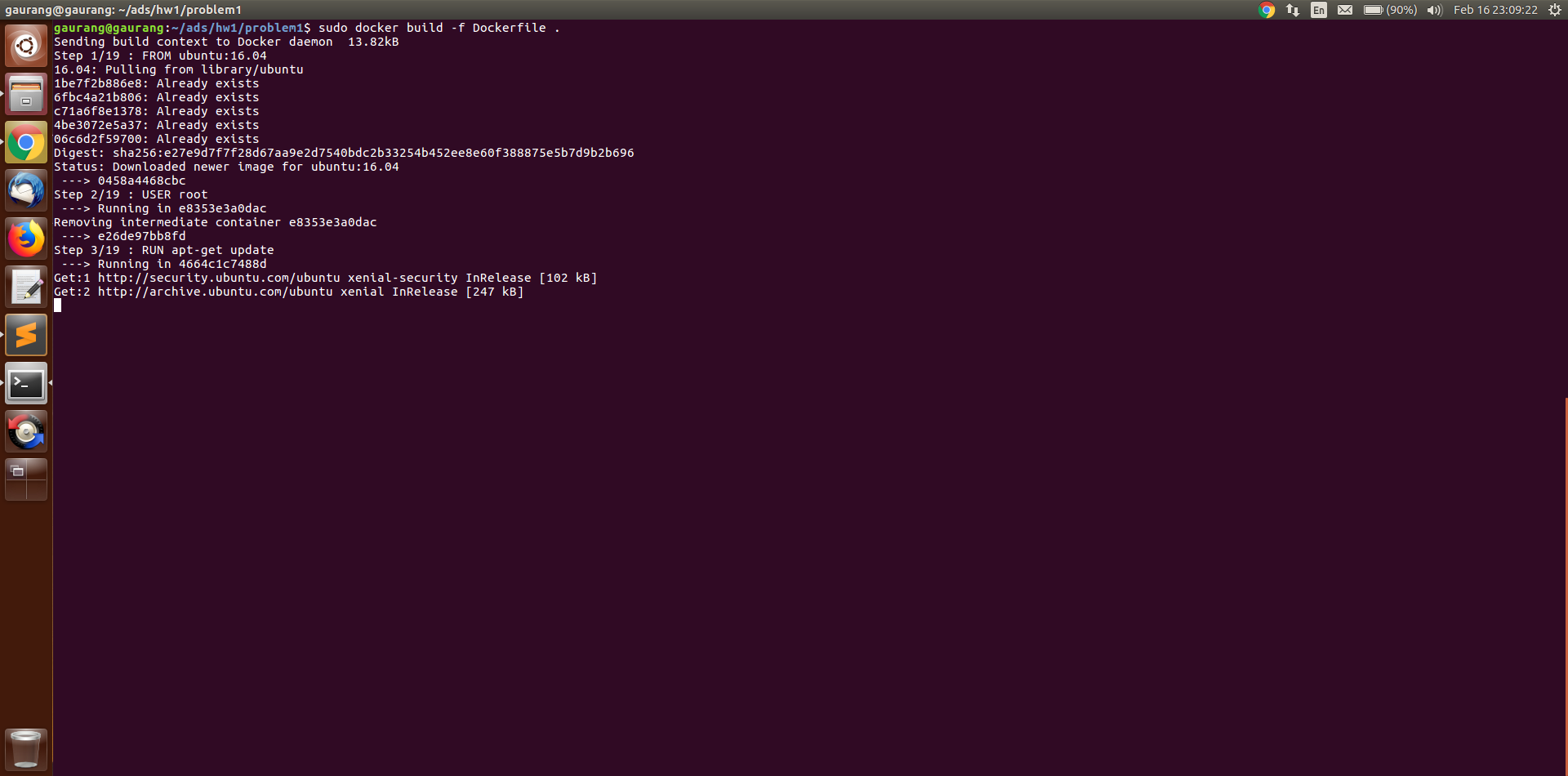
A log file is created to capture each task performed in Section 1 process along with timestamp which will make analysis at ease if any issue faced by a user. The file also logs the last point of failure based on the given inputs.



**Section 2 – Dockerize the pipeline**

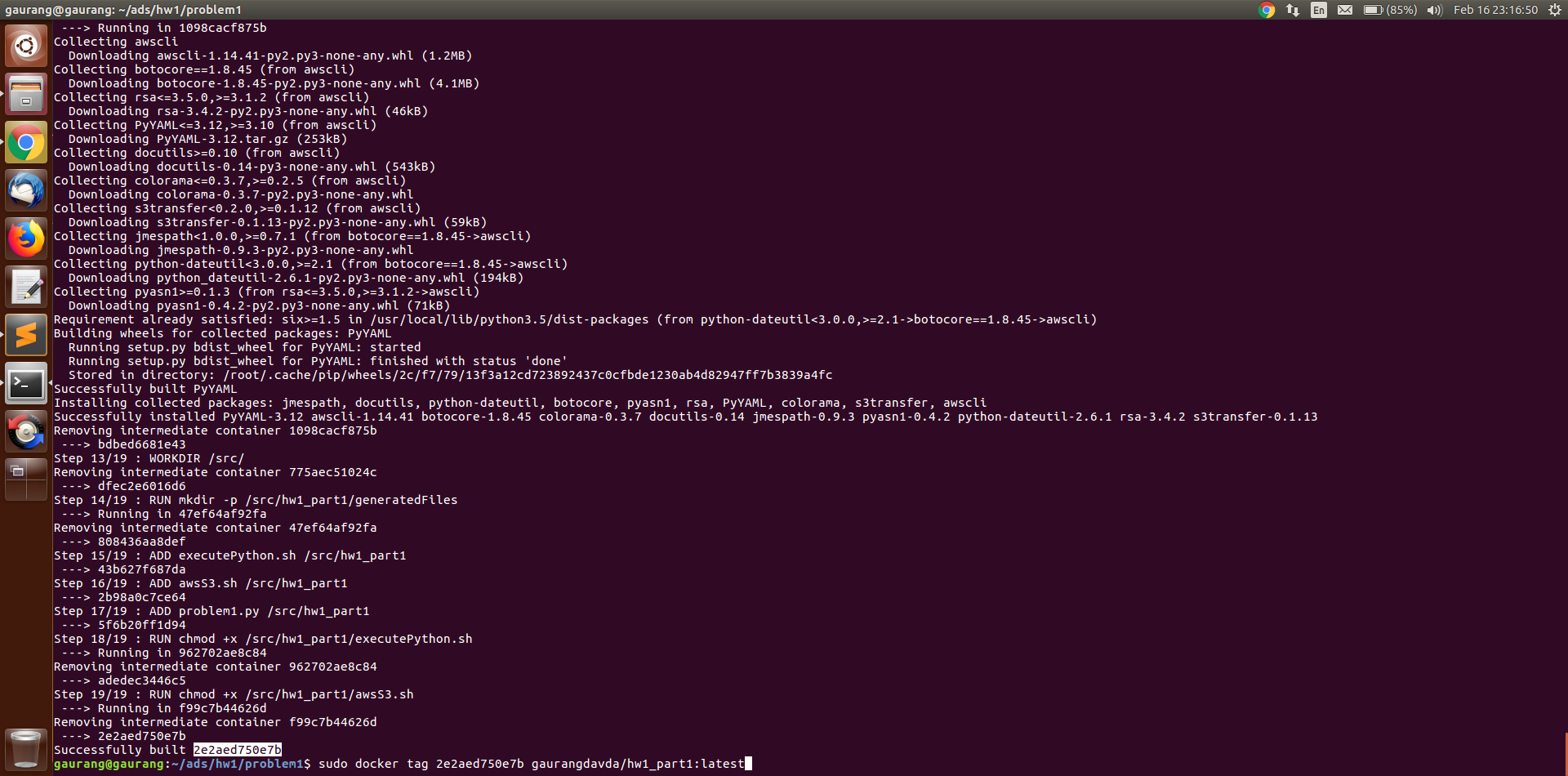
**STEP 1: Build the docker image from docker file**

sudo docker -f Dokerfile **.**



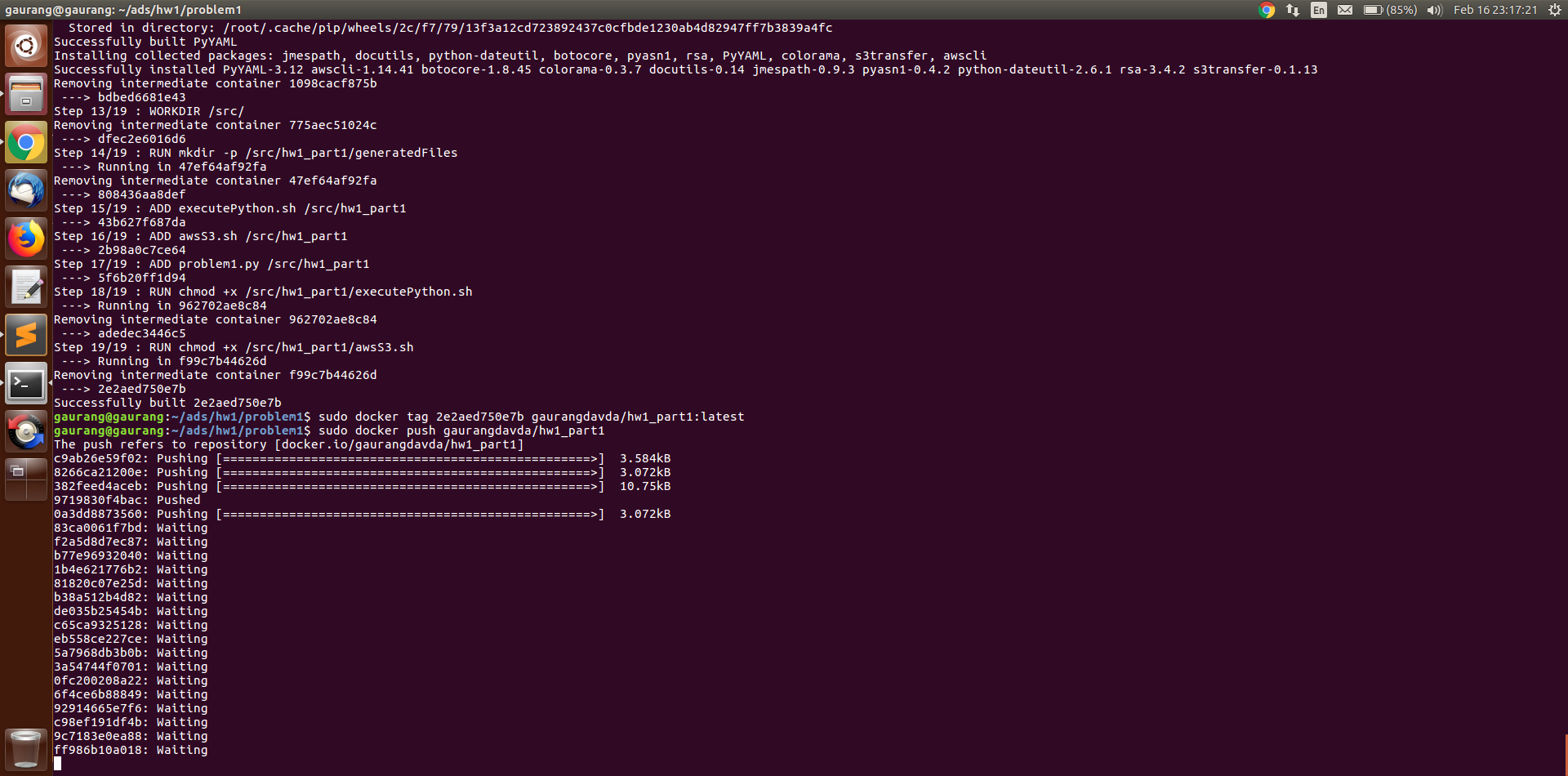
**STEP 2: Tag the build image to your repository name using**

sudo docker tag <tag of built file > <docker username >/<reponame>:<version>



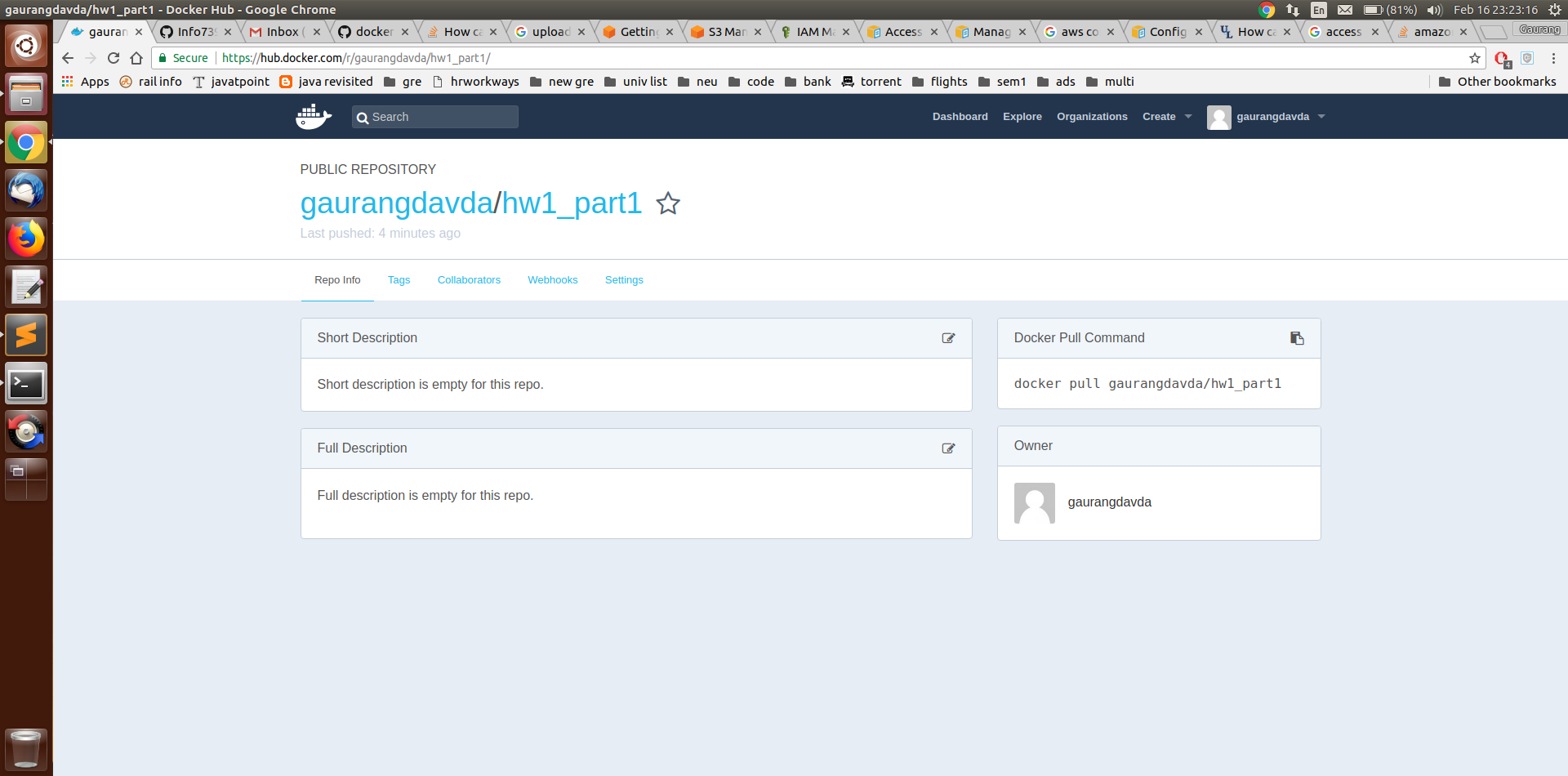
**STEP 3: Push the docker image from local to docker hub**

sudo docker push <docker username>/<reponame>



**STEP 4: Run the docker image using**

sudo docker run <docker username>/<reponame> <path of the file to be executed> <parameter of shell file>



**Section 3 – Discussing the output**

**How do you handle exceptions when you don’t find the CIK/accession number or if the amazon keys aren’t valid?**

In case, User did not provide the CIK/accession number, then the program will run for the default CIK/accession number which is this case is IBM.

In case, User provide the invalid CIK/accession number than the program will handle the error by displaying an error message (“Invalid URL”). The Log file are generated to capture the error.

There are three keys in amazon

1. Access key
2. Secret key
3. S3 path

The input parameters to the shell script are used and exported as environment variables in the docker container which will be used while configuring AWS . if any of these parameters are not there the shell script will throw an error at that point and will not proceed ahead

**Review your outputs stored on Amazon S3 and discuss outputs**

Amazon S3 will consist of a zip file which will have of all the csv files which are generated by our process through Docker. There will be one log file which contains all the process/function/error occurred during our execution of program.